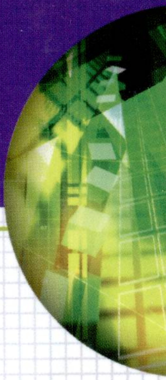
ELECTRONIC ENGINEERING

HARDWARE IMPLEMENTATION of FINITE-FIELD ARITHMETIC



JEAN-PIERRE DESCHAMPS
JOSÉ LUIS IMAÑA
GUSTAVO D. SUTTER

Hardware Implementation of Finite-Field Arithmetic

Jean-Pierre Deschamps
José Luis Imaña
Gustavo D. Sutter

McGraw Hill

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

The McGraw-Hill Companies

Cataloging-in-Publication Data is on file with the Library of Congress.

Hardware Implementation of Finite-Field Arithmetic

1 2 3 4 5 6 7 8 9 0 DOC/DOC 0 1 4 3 2 1 0 9

ISBN 978-0-07-154581-5
MHID 0-07-154581-6

The pages within this book were printed on acid-free paper.

Sponsoring Editor
Stephen S. Chapman

Acquisitions Coordinator
Alexis Richard

Editorial Supervisor
David E. Fogarty

Project Manager
Somya Rustagi

Copy Editor
Megha Roy Chaudhary

Proofreader
Barbara Danziger

Indexer
Jean-Pierre Deschamps

Production Supervisor
Richard C. Ruzycka

Composition
International Typesetting and Composition

Art Director, Cover
Jeff Weeks

Hardware Implementation of Finite-Field Arithmetic

About the Authors

Jean-Pierre Deschamps received an MS degree in electrical engineering from the University of Louvain, Belgium, in 1967, a PhD degree in computer science from the Autonomous University of Barcelona, Spain, in 1983, and a PhD degree in electrical engineering from the Polytechnic School of Lausanne, Switzerland, in 1984. He worked in several companies and universities. He is currently a professor at the University Rovira i Virgili, Tarragona, Spain. His research interests include ASIC and FPGA design, digital arithmetic, and cryptography. He is the author of seven books and about a hundred international papers.

José Luis Imaña received the MS and PhD degrees, both in Physics, from Complutense University of Madrid, Spain, where he is currently a professor. His research interests include algorithms and VLSI architectures for computations in finite fields, cryptography, computer arithmetic, reconfigurable computing architectures, and formal methods in verification. He is the author of about 30 international papers and communications.

Gustavo D. Sutter received an MS degree in computer science from State University UNCPBA of Tandil (Buenos Aires), Argentina, and a PhD degree from the Autonomous University of Madrid, Spain. He has been a professor at the UNCPBA, Argentina, and is currently a professor at the Autonomous University of Madrid, Spain. His research interests includes ASIC and FPGA design, digital arithmetic, and development of embedded systems. He is the author of one book and about 30 international papers and communications.

Preface

Finite fields are used in different types of computers and digital communication systems. Two well-known examples are error-correction codes and cryptography. The traditional way of implementing the corresponding algorithms is software, running on general-purpose processors or on digital-signal processors. Nevertheless, in some cases the time constraints cannot be met with instruction-set processors, and specific hardware must be considered, that is, circuits specifically designed for executing those complex algorithms: they implement the particular computation primitives of the algorithms and profit from their inherent parallelism.

Apart from the application-specific integrated circuits (ASICs) solution, another technology at hand for developing specific circuits is constituted by field-programmable gate arrays (FPGA). They form an attractive option for small production quantities as their nonrecurrent engineering costs are much lower than those corresponding to ASICs. They also offer flexibility and fast time-to-market. Furthermore, in order to reduce their size, and so the unit cost, an interesting possibility is to reconfigure them at run time so that the same programmable device can execute different predefined functions.

This book describes algorithms and circuits for executing the main finite-field operations, that is, addition, subtraction, multiplication, squaring, exponentiation, and division. It is mainly addressed to hardware engineers involved in the development of embedded systems, including finite-field operations. Distinguishing features of this book are the following:

- The emphasis is different from the classic texts on finite fields. It is not limited to the description of algebraic and algorithmic aspects. The main topic is circuit synthesis.
- A special importance has been given to FPGA implementations. The particular architecture of these components leads the designer to use synthesis techniques somewhat different than the ones applied for ASIC for which standard cell libraries exist. Throughout the book examples of FPGA implementation are described.

- Most algorithms are described in Ada, a programming language similar to VHDL, so that they can be executed and the correctness of the proposed algorithms can be verified with actual input data.
- In what concerns the description of the circuits, logic schemes are presented as well as VHDL models, in such a way that the corresponding circuits can be easily simulated and synthesized.

Overview

The book is divided into 10 chapters. The first chapter (mathematical background) gives the main definitions and properties of finite fields. Chapters 2 to 4 are dedicated to the operations modulo m and the corresponding circuits. Chapter 2 deals with the modulo m reduction, Chap. 3 with the modulo m addition, subtraction, multiplication, and exponentiation, and Chap. 4 with the modulo p division, where p is a prime. Chapters 5 and 6 are dedicated to the operations modulo $f(x)$, where $f(x)$ is a polynomial over a finite field, and to the corresponding circuits. Chapter 5 deals with the modulo $f(x)$ addition, subtraction, multiplication, and exponentiation, and Chap. 6 with the modulo $f(x)$ division, where $f(x)$ is an irreducible polynomial. Chapters 7 to 9 are dedicated to the main arithmetic operations over $GF(2^m)$. In Chap. 7 polynomial bases are considered (thus, a particular case of the topics dealt with in Chaps. 5 and 6). In Chap. 8 normal bases are used, and in Chap. 9 dual and triangular bases are considered. Chapter 10 is dedicated to elliptic-curve cryptography, currently one of the main finite-field applications.

There are four appendices. Three of them describe circuits for performing arithmetic operations over some particular fields, namely a prime field $GF(2^{192} - 2^{64} - 1)$ in App. A, two optimal extension fields $GF(239^{17})$ and $GF((2^{32} - 387)^6)$ in App. B, and two binary extension fields $GF(2^{163})$ and $GF(2^{233})$ in App. C. Appendix D is a brief comparison of the syntaxes of Ada and VHDL.

All the chapters, but the first one, include algorithms, circuits, and results of FPGA implementations. The algorithms are described in Ada and the circuits are modeled in VHDL. Complete and executable source files (Ada and VHDL) are available at the authors' Web site *www.arithmetic-circuits.org*.

Acknowledgments

The authors are grateful to the following universities for providing them the means for carrying this work through to a successful conclusion: University Rovira i Virgili (Tarragona, Spain), Autonomous University of Madrid (Spain), and Complutense University of Madrid (Spain).

Contents

CHAPTER 1

Mathematical Background

This chapter presents some topics in mathematics; it is intended to make this book self-contained. For further details the reader can refer to textbooks on Algebra ([Coh93], [GN03], [Her75], [Hun74]), Number Theory ([Kob94], [Ros92], [Ros00], [Gar59]), Finite Fields ([LN83], [LN94], [McC87], [Men93]), and Cryptography [MOV96], from where the following material has been mainly extracted.

1.1 Number Theory

1.1.1 Basic Definitions

Definitions 1.1

1. The set of natural numbers[1] $N = \{0, 1, 2, 3, \ldots\}$.
2. The set of integers $Z = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.

Definition 1.2 Given two integers x and y, y divides x (y is a *divisor* of x) if there exists an integer z such that $x = zy$.

Definition 1.3 Given two integers x and y, with $y > 0$, there exist two integers q (the *quotient*) and r (the *remainder*) such that

$$x = qy + r \qquad \text{where } 0 \le r < y$$

It can be proven that q and r are unique.
Then (notation)

$$r = x \bmod y \qquad q = x \operatorname{div} y$$

An alternative definition:

[1]For convenience, the element zero has been included in N.

Definition 1.4 (integer division) Given two integers x and y, with $y > 0$, there exist two integers q (the *quotient*) and r (the *remainder*) such that

$$x = qy + r \qquad \text{where } 0 \le r < y \text{ if } x \ge 0 \text{ and } -y < r \le 0 \text{ if } x < 0$$

It can be proven that q and r are unique. Then (notation)

$$r = x \text{ rem } y \qquad q = x/y$$

Examples 1.1

1. $x = -16, y = 3$:

$$-16 \text{ mod } 3 = 2, -16 \text{ div } 3 = -6, -16 = -6 \cdot 3 + 2$$
$$-16 \text{ rem } 3 = -1, -16/3 = -5, -16 = -5 \cdot 3 + (-1)$$

2. $x = -15, y = 3$:

$$-15 \text{ mod } 3 = 0, -15 \text{ div } 3 = -5, -15 = -5 \cdot 3 + 0$$
$$-15 \text{ rem } 3 = 0, -15/3 = -5, -15 = -5 \cdot 3 + 0$$

Definitions 1.5

1. Given two integers x and y, z is the *greatest common divisor* of x and y if
   z is a natural number (nonnegative integer),
   z divides both x and y,
   any other common divider of x and y is also a divider of z.

   Notation: $z = gcd(x, y)$.

2. Given two integers x and y, they are said to be *relatively prime* if $gcd(x, y) = 1$.

3. An integer $p > 1$ is said to be *prime* if its only positive divisors are 1 and p.

1.1.2 Euclidean Algorithms

Given two natural numbers x and y, the *Euclidean algorithm* for natural numbers computes $gcd(x, y)$. It is based on a series of integer divisions:

$$r(i-1) = q(i)r(i) + r(i+1) \qquad \text{where } 0 \le r(i+1) < r(i)$$

Observe that any divider of $r(i-1)$ and $r(i)$ is also a divider of $r(i)$ and $r(i+1)$ so that

$$gcd(r(i-1), r(i)) = gcd(r(i), r(i+1))$$

Initially

$$r(0) = x \qquad \text{and} \qquad r(1) = y$$

Then compute

$$\begin{aligned} r(0) &= q(1)r(1) + r(2) \\ r(1) &= q(2)r(2) + r(3) \\ r(2) &= q(3)r(3) + r(4) \\ &\dots \\ r(n-3) &= q(n-2)r(n-2) + r(n-1) \\ r(n-2) &= q(n-1)r(n-1) + r(n) \end{aligned}$$

where $r(1) > r(2) > \cdots > r(n) = 0$ and $gcd(r(i-1), r(i)) = gcd(r(i), r(i+1))$, so that

$$gcd(x, y) = gcd(r(0), r(1)) = \cdots = gcd(r(n-1), r(n)) = gcd(r(n-1), 0) = r(n-1)$$

Example 1.2 Let $r(0) = x = 9520; r(1) = y = 3120;$

$$\begin{aligned} 9520 &= 3.3120 + 160 \\ 3120 &= 19.160 + 80 \\ 160 &= 2.80 + 0 \end{aligned}$$

Then $gcd(9520, 3120) = 80$.

In the *extended Euclidean algorithm* a series of coefficients $b(i)$ and $c(i)$ is calculated in parallel with the computation of $r(0), r(1), r(2), \dots, r(n)$:

$$\begin{aligned} b(0) &= 1 & c(0) &= 0 \\ b(1) &= 0 & c(1) &= 1 \\ b(2) &= b(0) - b(1)q(1) & c(2) &= c(0) - c(1)q(1) \\ &\dots \\ b(n-1) &= b(n-3) - b(n-2)q(n-2) & c(n-1) &= c(n-3) - c(n-2)q(n-2) \end{aligned}$$

It can be demonstrated by induction that

$$r(i) = b(i)x + c(i)y \qquad \forall\, i = 0, 1, 2, \dots, n-1$$

In particular

$$gcd(x, y) = r(n-1) = b(n-1)x + c(n-1)y$$

In conclusion the extended Euclidean algorithm expresses the greatest common divisor z of two natural numbers x and y as a linear combination of x and y, that is,

$$z = bx + cy \tag{1.1}$$